IN THE SPECIFICATION

Please amend the paragraph bridging pages 9 and 10 as follows:

"The driver(s) may be configured to execute at the highest machine permission level. The taking step may include a step of freezing an operation of the operating system of the gaming machine. The taking step may also include a step of disabling interrupts on the gaming machine. The verifying step may include verifying a BIOS of a motherboard of the gaming machine. The verifying step may include verifying a BIOS of any add-on board within the gaming machine. The verifying step may include verifying ROM shadowing within the gaming machine, verifying hardware registers, verifying a signature in memory of the at least one driver, verifying the content of files on disk within the gaming machine and/or verifying the downloadable micro-code of smart hardware within the gaming machine, for example. The method may further include a step of auditing the source code of the driver(s) by a third party. The source code of the driver(s) may also be audited by a game certification lab. The method may further include a step of certifying the driver(s) by a game certification lab and/or by a third party. The gaming machine may be controlled by a PC, the driver(s) may be code signed and the installing step may be triggered by one or more plug-and-play dongles inserted in one or more ports of the PC. The driver(s) installed in the installing step may be code-signed by Microsoft's WHQL - or another certifying agency, for example. The verifying step may verify the legitimacy of the software and memory contents without modifying the content thereof and the method further may include a step of reporting an outcome of the verifying step. The gaming machine further may include a third party dongle installed therein and the driver(s) may be linked to the third party dongle to enable the third party to audit the driver(s). The gaming machine further may include a hard disk drive that may include a partition formatted for simple file access (by means of a **File Allocation Table (FAT)** ~~FAT~~, for example) and wherein the method further may include a step of accessing code-signed downloaded software from the simple file access partitioned hard disk drive. The hard disk drive partition may be formatted according to FAT2 protocol, for example. The verifying step may verify the memory content stored on one or more of the following within the gaming machine: a hard disk drive of the gaming machine, an optical memory of the gaming machine, flash memory of the gaming machine, non-volatile **Random Access Memory (RAM)** ~~RAM~~ memory of the gaming machine, ferromagnetic memory of the gaming machine, magnetic memory of the gaming machine, and/or holographic memory of the gaming machine, for example.

Please amend the second full paragraph of page 4 as follows:

The most promising approach available today in a COTS multimedia product that offers comprehensive security for preventing unauthorized code from executing, is integrated in Microsoft Windows XP, Windows 2000 and Windows .NET. There are three technologies that address three different layers; namely, (1) Driver Signing, (2) Windows File Protection and (3) Software Restriction Policies. These three technologies cover all but two aspects of possible execution by unauthorized modified software code, that is, (1) by modification of the motherboard **Basic Input Output System (BIOS)** ~~BIOS~~ or other add-on boards such as a graphic card with on-board BIOS or a **Small Computer System Interface (SCSI)** ~~SCSI~~ controller with dedicated on-board BIOS, (2) by modification of an emulated CPU such as downloadable microcode for the Transmeta microprocessor that emulates Intel CPU instructions. The risk with the emulated CPU instructions can be simply avoided by not allowing the use of such emulating microprocessors. It is, therefore, another object of this invention to provide a trusted mechanism to verify that the motherboard BIOS and add-on BIOS are not unauthorized. It is a further object of this invention to provide a trusted mechanism to verify memory content, hardware register content and any form of data storage media. Verification, according to embodiments of the present invention, relies on a hash signature or on code signing with a trusted certificate.

Please amend the paragraph bridging pages 9 and 10 as follows:

The driver(s) may be configured to execute at the highest machine permission level. The taking step may include a step of freezing an operation of the operating system of the gaming machine. The taking step may also include a step of disabling interrupts on the gaming machine. The verifying step may include verifying a BIOS of a motherboard of the gaming machine. The verifying step may include verifying a BIOS of any add-on board within the gaming machine. The verifying step may include verifying **Read Only Memory (ROM)** ~~ROM~~ shadowing within the gaming machine, verifying hardware registers, verifying a signature in memory of the at least one driver, verifying the content of files on disk within the gaming machine and/or verifying the downloadable micro-code of smart hardware within the gaming machine, for example. The method may further include a step of auditing the source code of the driver(s) by a third party. The source code of the driver(s) may also be audited by a game certification lab. The method may further include a step of certifying the driver(s) by a game certification lab and/or by a third party. The gaming machine may be controlled by a PC, the driver(s) may be code signed and the installing step may be triggered by one or more plug-and-play dongles inserted in one or more ports of the PC. The driver(s) installed in the installing step may be code-signed by Microsoft's

WHQL - or another certifying agency, for example. The verifying step may verify the legitimacy of the software and memory contents without modifying the content thereof and the method further may include a step of reporting an outcome of the verifying step. The gaming machine further may include a third party dongle installed therein and the driver(s) may be linked to the third party dongle to enable the third party to audit the driver(s). The gaming machine further may include a hard disk drive that may include a partition formatted for simple file access (by means of a FAT, for example) and wherein the method further may include a step of accessing code-signed downloaded software from the simple file access partitioned hard disk drive. The hard disk drive partition may be formatted according to **File Allocation Table 2 (FAT2)** ~~FAT2~~ protocol, for example. The verifying step may verify the memory content stored on one or more of the following within the gaming machine: a hard disk drive of the gaming machine, an optical memory of the gaming machine, flash memory of the gaming machine, non-volatile RAM memory of the gaming machine, ferromagnetic memory of the gaming machine, magnetic memory of the gaming machine, and/or holographic memory of the gaming machine, for example.

Please amend the second paragraph of page 1 as follows:

On-line download of updated software and new games has been performed routinely with lottery terminals since the on-line capture of lottery slips started to be deployed in the late 1980s. The techniques and procedures have been refined along the years and are now considered as essential features. On the other hand, casino regulators have always been reluctant to introduce on-line download of updated software and of new games for casino gaming machines. Such reluctance stems from concerns relative to unauthorized intrusion and malicious modification of software code. These concerns are understandable, particularly since the late 1990s because of the general trend of constructing gaming terminals using standard **Personal Computer (PC)** ~~PC~~ hardware and PC software platforms that are subject to assault by hackers that are well versed in the techniques for taking advantage of the known weaknesses and flaws of such platforms. Even now with lotteries, the appeal of making use of the broadband public Internet network instead of private networking is considerable, but there are indeed significant security concerns and consequently new plans are blurred with uncertainty.

Please amend the paragraph bridging pages 3 and 4 as follows:

There is no better alternative for casinos and lotteries gaming computer hardware but to adopt standard PC hardware controlled by the latest generation multimedia software from Microsoft, QNX (**a commercial Unix-like real-time operating system, aimed primarily at the embedded systems market**), WindRiver Systems, Unix or from the Linux community. It is, therefore, an object of this invention to provide additional security mechanisms that can perform independent and trusted verification of the Commercial-Off-The-Shelf (COTS) software installed on the gaming terminals that can be trusted because of its precisely defined objectives and the availability of source code for peer review and certification by gaming certification labs.

Please amend the second full paragraph of page 32 as follows:

The process flow 1000 differs from process flow 900 in that the Trusted Verification driver performs a verification of the operating system components 1032 against a trusted reference. In order for the Trusted Verifier driver to be able to verify the operating system components, necessary access mechanisms to the files must be available. Software to access files on **File Allocation Table 16 (FAT16)** ~~FAT16~~ or **File Allocation Table 32 (FAT32)** ~~FAT32~~ formatted disk partitions is quite common. Software to access files on advanced disk partitions such as Microsoft **New Technology File System (NTFS)** ~~NTFS~~ is less common. Examples of third party products that are capable of accessing NTFS files independently of Microsoft Windows operating system are Partition Magic from PowerQuest Corp. www.powerquest.com and Partition Commander from V Communications, Inc. (www.v-com.com). Source code for allowing NTFS file access is available on the Internet from various freelance developers. In addition, Microsoft is making available the source of its operating system to selected developers.

Please amend the fourth paragraph of page 21 as follows:

It is not clear at this stage to what level Palladium extends as suggested at 632 ~~and 633~~, but it is likely that this will at least bridge the gap with the Driver Signing layer 620. The Palladium software code 630 cooperates with the security devices buried within the microprocessor and other secure devices embedded on the computer board to provide a trusted base for everything that executes on higher levels.

Please amend the first full paragraph of page 17 as follows:

There is a dilemma in the code-signing verification process 300, however, in that the process itself might be a fraudulent verification process. Consequently, it is a necessary to be able to verify that the verification platform can be trusted. The code verification processes 128 and 140 may advantageously be replaced by the process according to the present invention, as shown in Fig. 4. The code-signing verification 400 starts at 402 by verifying that the code-signing verification platform can be trusted, as shown at 404, 410. If not **as shown at 406**, then an alert 408 is raised and the overall process fails. If trust can be established as shown at 410, then the code-signing verification can be safely executed, as indicated at 412. If the code-signing verification detects an anomaly as shown at 414, 416, then an alert 418 is raised and the overall process fails. If the code-signing verification succeeds at 420, then the process returns 422 to the main flow 100 as shown in Fig. 1.

Please amend the first full paragraph of page 15 as follows:

Code signing may be accomplished as shown in Fig. 2 **at 200**. The signing utility uses a hash algorithm 212 on the executable code 202, 210 to compute a digest 216 (which is also known as a one-way hash) by securely compressing executable code 202 of arbitrary length into a fixed-length digest result 216 **via 214**. The most common hash function algorithms used in code signing are the Secure Hash Algorithm (SHA), Message Digest Algorithm 4 (MD4), or MD5. The resulting length of the digest is a function of the hash function algorithm, but a common digest length is 128 bits. The digest 216, 218 is then encrypted 220 using the trustee's private key 222, 224. The encrypted digest 226, 228 and the trustee's digital certificate 230, 232, 234 are then appended to the executable code 202,204, 208 to form the signed code 206. The certificate 230, 234 contains the trustee's public key 231. **Reference 236 shows the code signer's private key 222 and Certificate 230, as shown in Fig. 2.**

Please amend the second full paragraph of page 16 as follows:

2. Step 2 (318): If it is, the certificate 304 identifies the hash function algorithm 212 that was used to create the signed digest 216 within the received signed code 206, 302. With this information, the same hash algorithm code 320 that was used to create the original digest 216 is then applied to the received executable code 310, 312, 314 (via 316), creating a digest value 322, 324, which then is temporarily stored.

After the first (one sentence) paragraph of page 21, insert the following paragraph:

A simplified layered view of the Microsoft security model can be examined on the diagram shown at 600 in Fig.6. The computer hardware 602 is controlled directly via the motherboard BIOS 604, the add-on card BIOS 606, the Hardware Abstraction Layer (HAL) 612 and the DirectX 616 services. The motherboard BIOS 604 has interfaces with the drivers 608 and the HAL 612. The operating system kernel 610 has interfaces with the drivers 608 and the HAL 612 on the lower side, and to the OS services 614 on the upper side. The gaming applications 618 reside on top of the OS services 614.

The Software Restriction Policies technology 624 ensures that only code signed by trusted parties can execute. The code forming the Software Restriction Policies platform is embedded within the operating system and it can be trusted to execute because the Windows File Protection technology 622 ensures that the code is unmodified.

Please amend the paragraph bridging pages 23 and 24 as follows:

A trusted mechanism for verifying the code signing of downloaded game software in a gaming machine according to an embodiment of the present invention is represented in Fig. 7 **at 700**. The various elements **712-720** shown in Fig. 7 that bear the same **non-numerical** label correspond to the identically labeled elements in Fig. 6 and the description thereof is omitted here for the sake of brevity. Fig. 7 includes, however, a driver named "trusted verifier", referenced at numeral 702. Drivers are a special class of software components that are capable of accessing the totality of the hardware resources 710 of the computer. When provided by third parties for controlling the add-on hardware that they sell that can be added to the computer, such as a SCSI hard disk controller and a graphics card for example, the third party drivers (a part of 704) are notorious for creating system instabilities and crashes. Furthermore, drivers may introduce fraudulent code that cannot easily be detected or protected against. Fortunately, "script kiddies" that are notorious for releasing countless variants of viruses on the Internet generally do not have the specific knowledge required to develop new "driver viruses". However, a very determined software developer specialized in the coding of drivers may at any time take advantage of this latent opportunity. The same applies to the motherboard BIOS 706 and the add-on board BIOS 708 (one or a plurality of add-on boards and their associated BIOS), especially BIOS stored in Flash memory that can be downloaded from the Internet, or BIOS that is copied from slow access ROM memory to fast RAM (this technique is known as "ROM shadowing"). Nowadays, the BIOS for the motherboard and add-on boards, as well as the firmware for hard disk drives, CD-ROM Writers, and other intelligent peripheral devices can be updated, either manually or automatically, using software code downloaded from the Internet.

Please amend the first full paragraph of page 31 as follows:

If all the verifications are successful at 1014, 1022, 1030, this indicates that the lower components of the software platform and of the hardware platform are trusted and that consequently, higher-level secure verification can be trusted. A process may be executed to verify whether the operating system components 1032 can be trusted. **If not, as shown at 1034, an alert 1036 may trigger.** ~~This~~ **The determination whether the operating system components 1032 may be trusted** may be done by accessing the operating system files on the system storage media and by verifying their hash or code-signature with certificate against a trusted reference. Success at 1038 indicates that the operating system can be trusted, as no unauthorized modification has been detected.

Please amend the paragraph bridging pages 32 and 33 as follows:

A preferred embodiment of the invention may use Microsoft Windows Hardware Quality Lab (WHQL) scheme 1000 depicted in Fig. 11 **at 1100**. As shown, the method starts at 1102 and the vendor or developer submits the driver executable code and auxiliary data to Microsoft WHQL at 1104. The Microsoft WHQL performs driver code analysis and testing at 1006 to verify the conformity of the driver's code with a set of rules. If the testing 1108 fails at 1110, the software is returned to the vendor at 1112, along with the test reports. If, however, the WHQL testing is successful as shown at 1114 then the driver is code-signed with a Microsoft Digital Signature at 1116. The code-signed driver is sent to the vendor/developer or alternatively is published on the Windows Update server at 1118 for any user connected to Internet to access through the Microsoft Windows Update technology. **The method ends at 1120.**